

**Amendments to Specification:**

Please update the paragraph beginning at line 13 through line 20, page 3, as follows:

Different methods have been proposed for carrying out modular multiplication. In particular, in an article appearing in "The Mathematics of Computation," Vol. 44, No. 170, April [1995] 1985, pp. 519-521, Peter L. Montgomery describes an algorithm for "Modular Multiplication without Trial Division." However, this article describes operations that are impractical to implement in hardware for a large value of  $N$ . Furthermore, the method described by Montgomery operates only in a single phase. In contrast, the system and method presented herein partitions operational cycles into two phases. From a hardware perspective, the partitioning provides a mechanism for hardware sharing which provides significant advantages.

Please update the paragraph beginning at line 17 through line 18, page 12, as follows:

Figure 3 is a block diagram similar to Figures 1 and 2 except more particularly showing those data flow paths which are active during the second or Z-phase of calculation[.];

Please update the paragraph beginning at line 4 through line 5, page 14, as follows:

Figure 13 (depicted as Figures 13A and 13B in the drawings) is a block diagram illustrating an improved rightmost processor element in which an adder in a critical path has been moved to improve performance;

Please update the paragraph beginning at line 15 through line 20, page 15, as follows:

The structure and operation of the present invention is dependent upon the partitioning of one of the multiplying factors into a plurality of  $k$  bit-wide pieces. Thus, instead of representing a binary number  $A$  as  $\sum_{i=0}^{n-1} a_i 2^i$ , one of the multiplying factors in the present invention is represented instead in the form  $A_{m-1} R^{m-1} + \dots + A_2 R^2 + A_1 R + A_0 = \sum_{j=0}^{m-1} A_j [R^j] \underline{R}$ , where  $R = 2^k$ . In this representation, the number  $A$  is represented in block form where each of the  $m$  blocks includes  $k$  bits. That is, each  $A_i$  represents an integer having  $k$  bits.